Quick Start Guide

For
openCONFIGURATOR

[Ethernet POWERLINK Configuration Tool]

Prepared By
Kalycito Infotech Pvt Ltd.,
India

Identifier	Quick_Start_Guide	**Version**	1.02
Prepared By	Ramakrishnan P	**Date**	05-Jul-2013
Approved By	Vinod PA	**Confidentiality**	Public domain document

Revision History

Version	Date	Modified By	Remarks
0.01	15-Apr-2009	Kalycito Powerlink Team	Initial Draft
1.00	18-May-2009	Kalycito Powerlink Team	Finalized and has cosmetic changes
1.01	29-Apr-2013	Ramakrishnan P	Updated the guide with the new features and major changes to the document for the openCONFIGURATOR version 1.3.0
1.02	05-Jul-2013	Ramakrishnan P	Minor changes and updated section 3.4

Table of Contents

1 Introduction....5
1.1 Context....5
1.2 Scope....5
1.3 References....5
2 Setup....6
2.1 Install....6
2.1.1 Linux....6
2.1.2 Windows (XP, Vista & 7)....6
2.2 Launch....9
2.2.1 Linux....9
2.2.2 Windows....9
3 Sample project....10
3.1 Creating a sample project....10
3.2 Add a CN....14
3.3 Adding process variable to openPOWERLINK CN....17
3.4 PDO Mapping for the CN process variables....24
3.5 Build the sample project....28
4 Conclusion....32
5 Appendix – Abbreviations....33
6 Support....34

Illustration Index

Illustration 1: Installer - License page....6
Illustration 2: Installer - Components Page....7
Illustration 3: Installer - Install Path....7
Illustration 4: Installer - Start Menu....8
Illustration 5: Installer - Install Completed....8
Illustration 6: Windows - Launch Tool....9
Illustration 7: New Project Menu....10
Illustration 8: Create New Project....10
Illustration 9: Project Wizard - Name....11
Illustration 10: Project Wizard – MN XDD....12
Illustration 11: MN Node Created....13
Illustration 12: Add CN Menu....14
Illustration 13: Add CN Window....15
Illustration 14: CN Created....15
Illustration 15: View Menu....16
Illustration 16: Add Index Menu....17
Illustration 17: Add Index Window....17
Illustration 18: Index Added - Tree....18
Illustration 19: Index Properties - Empty....19
Illustration 20: Add Index Properties....20
Illustration 21: Index Added....20
Illustration 22: Add SubIndex Menu....21
Illustration 23: Add SubIndex Window....21
Illustration 24: Add SubIndex Properties- 1....22
Illustration 25: Add SubIndex Properties- 2....22
Illustration 26: Input PI variables....23
Illustration 27: PDO mapping table....24
Illustration 28: PDO mapping table - Select Node id....25
Illustration 29: PDO mapping table - Select Index Id....25
Illustration 30: PDO mapping table - Select SubIndex Id....25
Illustration 31: PDO mapping table - Select Length....26
Illustration 32: PDO mapping table - Offset....26
Illustration 33: PDO mapping table - RPDO mapping values....27
Illustration 34: Build Project Menu....28
Illustration 35: Build Project - Auto Generate....28
Illustration 36: cdc_xap Folder View....29

1 Introduction

1.1 Context

The objective of this document is to demonstrate to users how openCONFIGURATOR can be installed and used to create and edit the network configuration by creating a sample project with a CN having one RPDO and one TPDO.

1.2 Scope

This document limits its scope with explaining how to create the demo project using openCONFIGURATOR tool.

1.3 References

- openCONFIGURATOR User Manual Document Version 1.3.0
- Ethernet POWERLINK Communication Profile Specification 301 Version 1.1.0

2 Setup

Download latest version of openCONFIGURATOR from http://sourceforge.net/projects/openconf/

2.1 Install

2.1.1 Linux

- Un-tar the openCONFIGURATOR.tar.gz file
- Open the terminal, and move to the extracted directory
- To check & install the required packages, run '*sudo ./configure*'
- If configuration succeeds, Makefile will be created
- To install openCONFIGURATOR, run '*sudo make install*' from the terminal

2.1.2 Windows (XP, Vista & 7)

- For Windows(XP, Vista & 7), please install the ActiveTCL version 8.5.14. The executable can be obtained from http://www.activestate.com/activetcl/downloads
- Unzip the openCONFIGURATOR.zip file
  - Windows XP
    - Run the openCONFIGURATOR_Setup file and follow the instructions
  - Windows Vista & 7
    - Run the openCONFIGURATOR_Setup file as Administrator [right click on the setup file and click on 'Run as Administrator'] and follow the instructions
- Now the Installer Dialog will open as shown below
- Read through the License and if you agree, press **'I Agree'** button and proceed with the installation

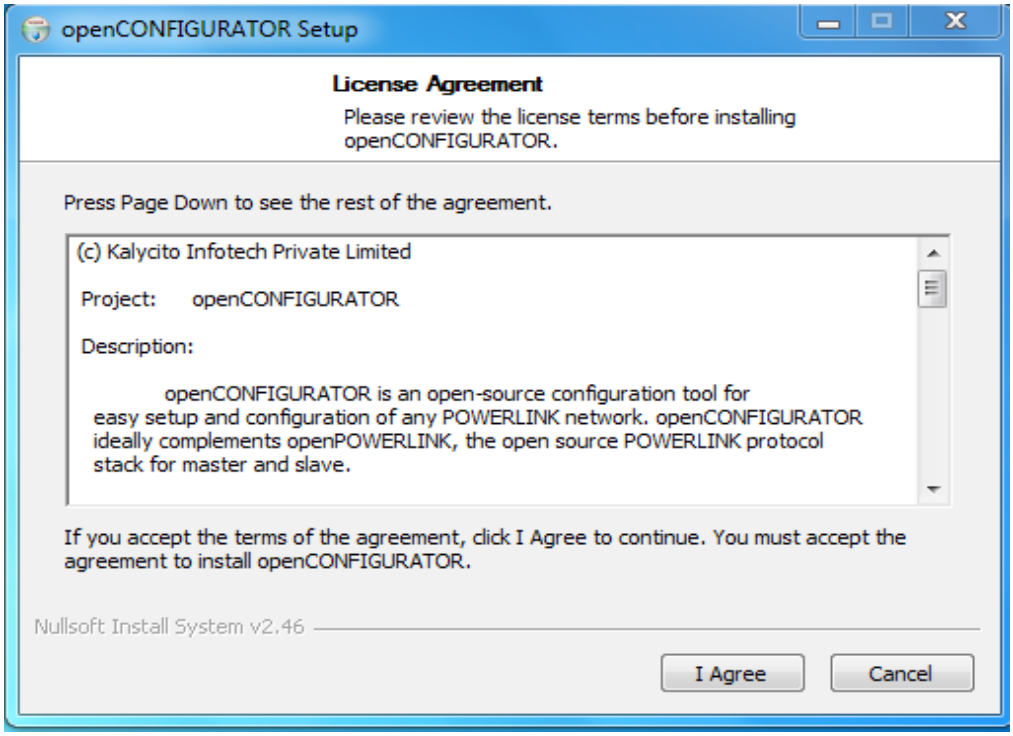


Illustration 1: Installer - License page

- Click '**Next**'

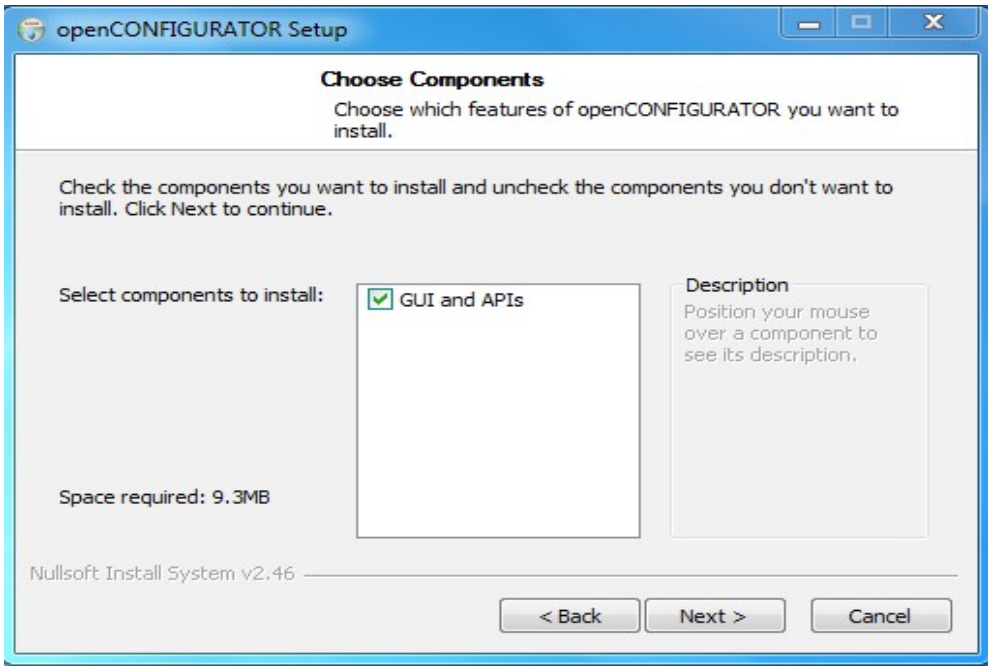


Illustration 2: Installer - Components Page

- Select the directory where the tool should be installed. Click '**Next**'

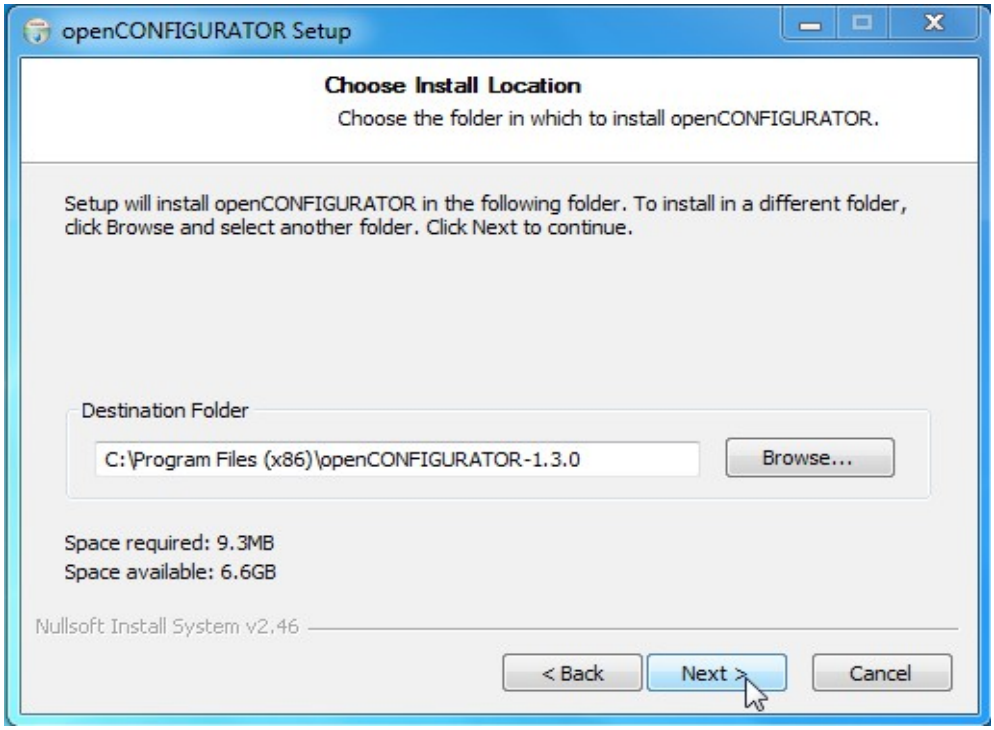


Illustration 3: Installer - Install Path

- Click '**Install**'

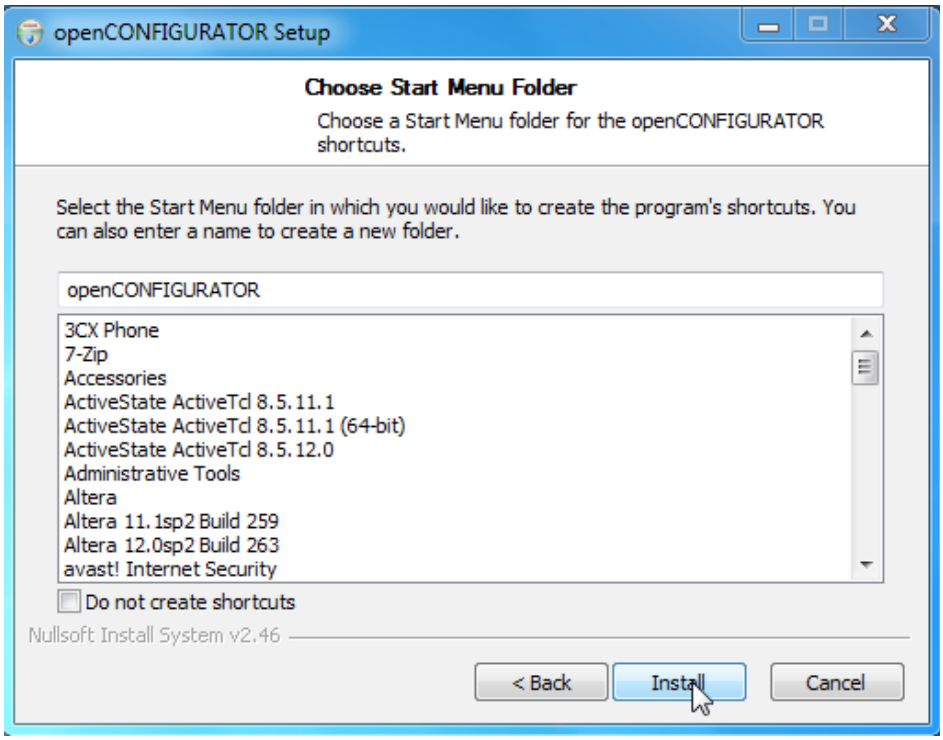


Illustration 4: Installer - Start Menu

- Now the installation is completed successfully

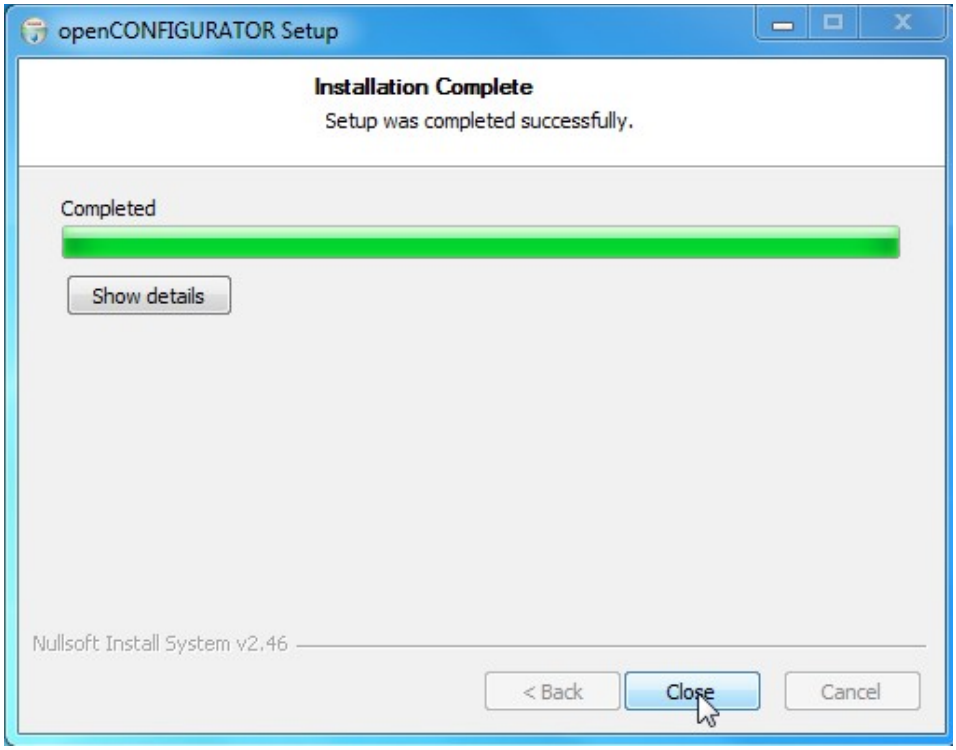


Illustration 5: Installer - Install Completed

2.2 Launch

2.2.1 Linux

From command prompt:

- Open the terminal
- Type openCONFIGURATOR and hit enter

From GUI:

- Go to Applications > Programming
- Click on 'openCONFIGURATOR'

2.2.2 Windows

- Go to Start Menu > All Programs > openCONFIGURATOR
- Click on openCONFIGURATOR

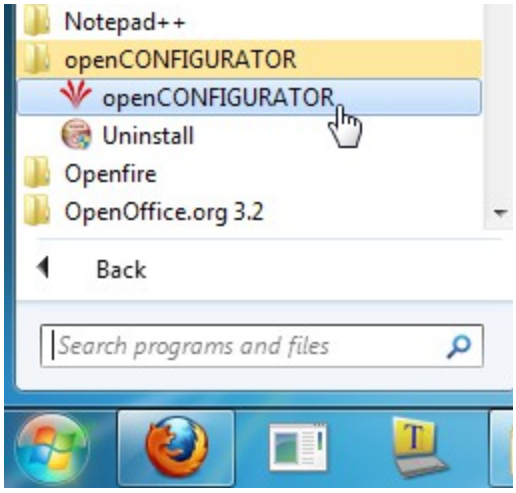


Illustration 6: Windows - Launch Tool

3 Sample project

This sample project will be used to create a network configuration having one CN with one byte TPDO & one byte RPDO mapped to the MN.

3.1 Creating a sample project

- The users can create a new project by selecting the 'File > New Project' or using 'CTRL + N' as the keyboard shortcut

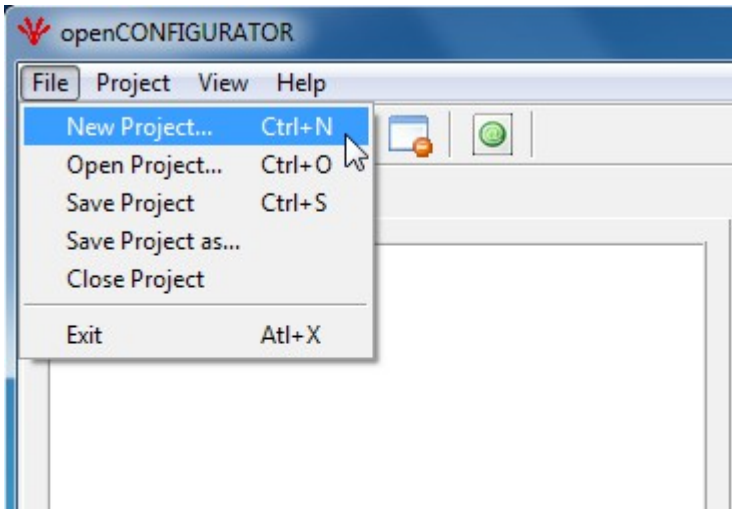


Illustration 7: New Project Menu

- To create a project, choose 'Create New Project' option and click 'OK'

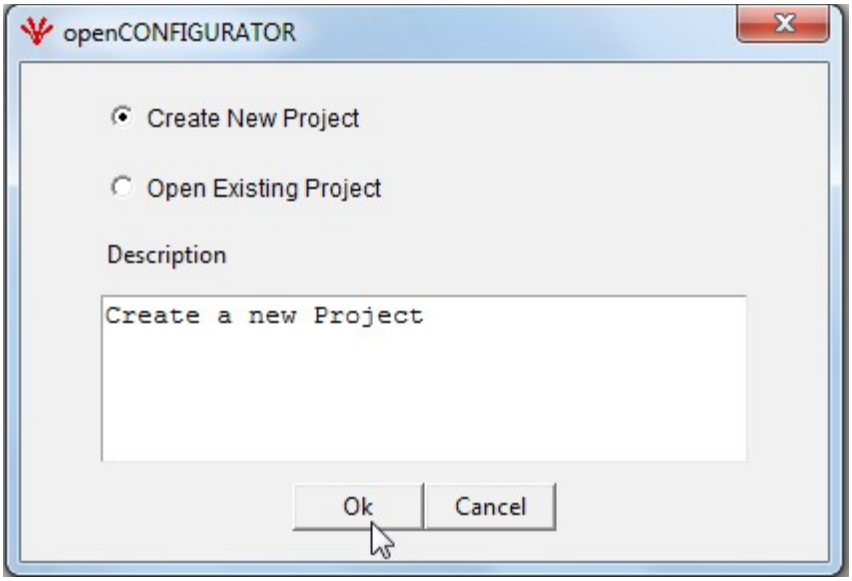


Illustration 8: Create New Project

- After clicking on the 'Ok' button the tool will guide you through a project creation wizard
  - Set the 'Project Name' for the project
  - Choose the project location by clicking on 'Browse'
  - Choose the 'Save' option as required

Save option	Description
AutoSave	Saves the configuration automatically without prompting the user
Prompt	Prompts the user for changes to be saved or not
Discard	Discards any modifications made to the configuration

- Here, we choose Sample as the 'Project Name' and 'Prompt' as the save option and click 'Next'

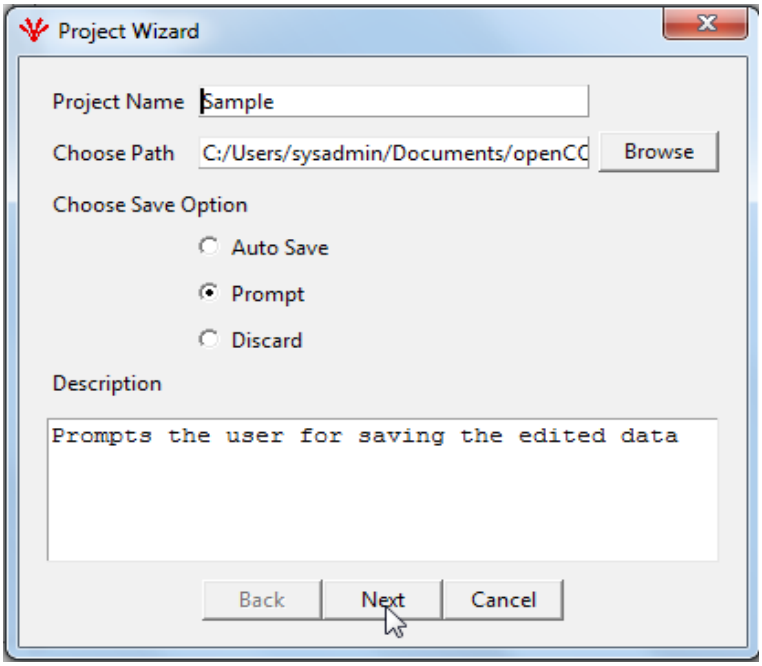


Illustration 9: Project Wizard - Name

- After clicking on the “Next” button the tool will guide you through the next step of the project creation wizard
  - MN configuration

Configuration Option	Description
Default	Default MN xdd which will be available with the installation package
Import XDD/XDC	User defined MN configuration

  - Auto Generate option

Auto Generate Option	Description
Yes	The MN configuration will be auto generated with the available CN's configuration
No	The MN configuration will have to be manually generated/updated by the user

- Here, we choose the 'default' MN XDD and 'Yes' as Auto Generate option and then click 'Ok'.

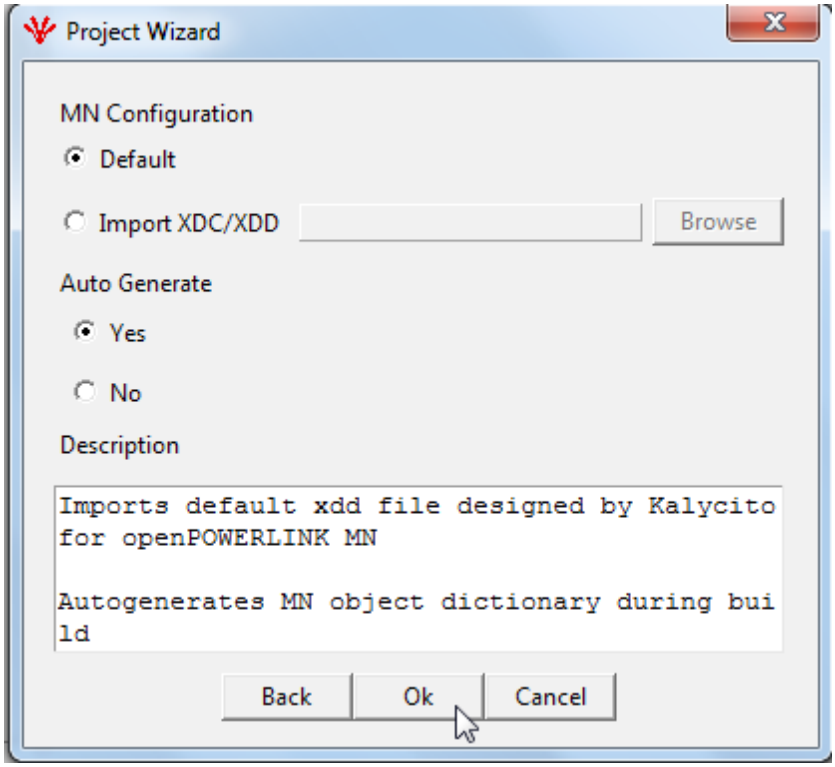


Illustration 10: Project Wizard – MN XDD

- Now the MN node will be created and the project window will look similar to the below illustration

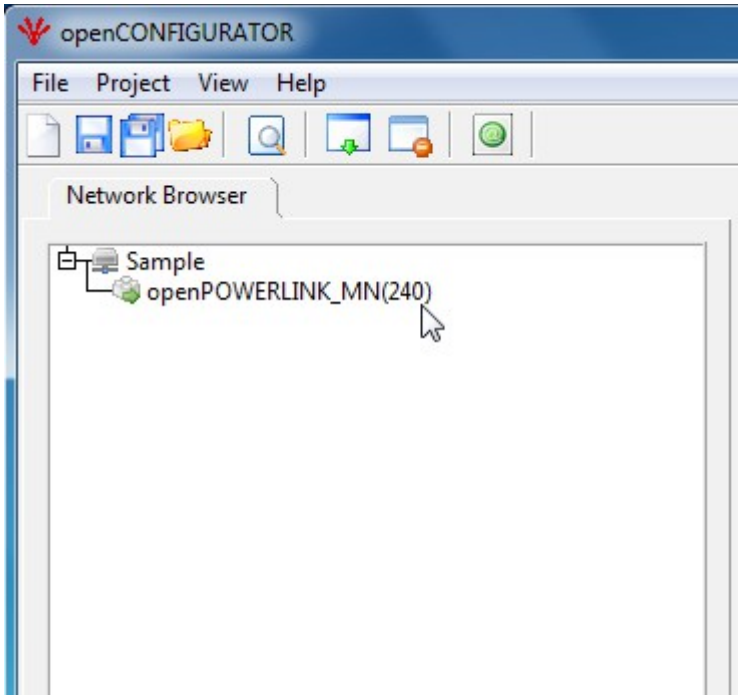


Illustration 11: MN Node Created

3.2 Add a CN

- Then to add the Controlled Node, right click on openPOWERLINK_MN and choose 'Add CN' option.

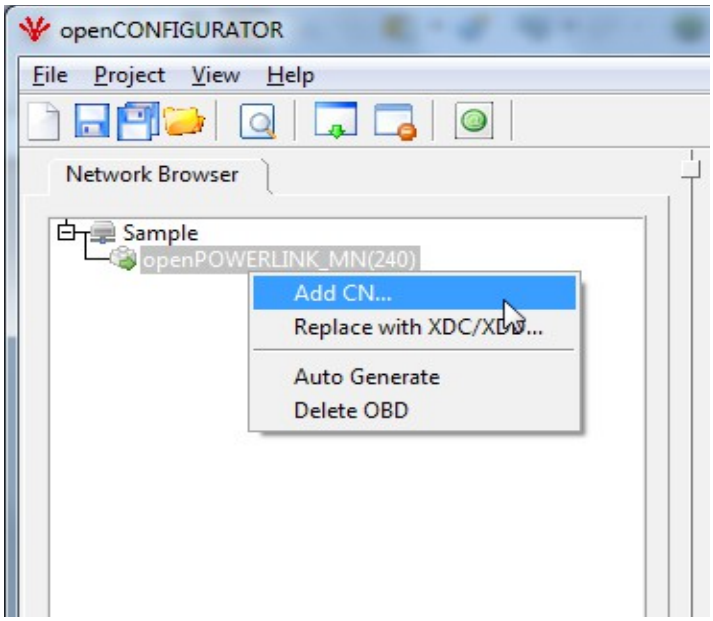


Illustration 12: Add CN Menu

- Now the 'Add New Node' dialog box will pop up prompting the user to give the configuration for the CN. Choose from the below table and click 'Ok'

New Node Configuration	Description
Name	Name for the Node
Node ID	Node Id for the Node. Range(1 - 239)
CN Configuration: Default	Default CN xdd which will be available with the installation package
CN Configuration: Import XDD / XDC	User defined configuration for the CN

- Here, we choose Name as 'CN_1' and NodeID as '1' and 'Default' CN xdd as the configuration

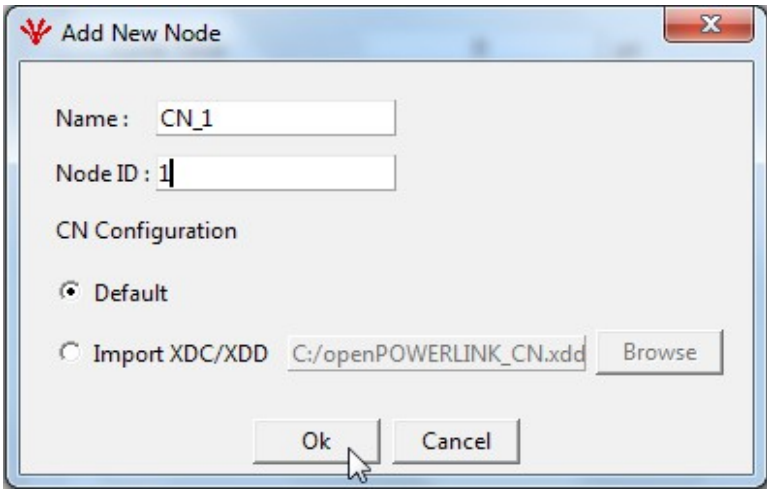


Illustration 13: Add CN Window

Warning	If you choose to import your XDD, please validate your XDD with the free XDD-Check utility available through the website of the Ethernet Powerlink Standardization Group (http://www.ethernet-powerlink.org) before importing it into openCONFIGURATOR

- After adding a CN the project window will look similar to the below Illustration

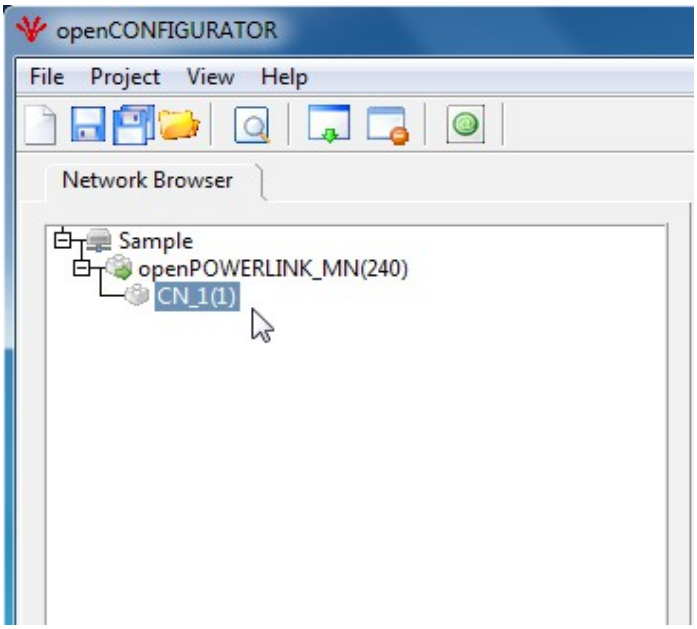


Illustration 14: CN Created

- Select View > Advanced View to add the PDO mapping to the CN

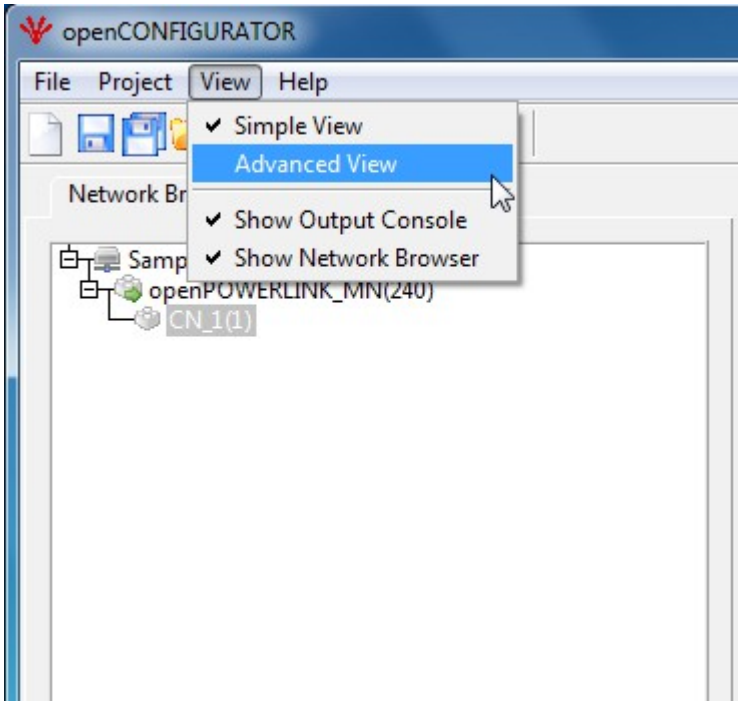


Illustration 15: View Menu

3.3 Adding process variable to openPOWERLINK CN

- To add an output process variable index to a CN, right click on the CN and select 'Add Index' option

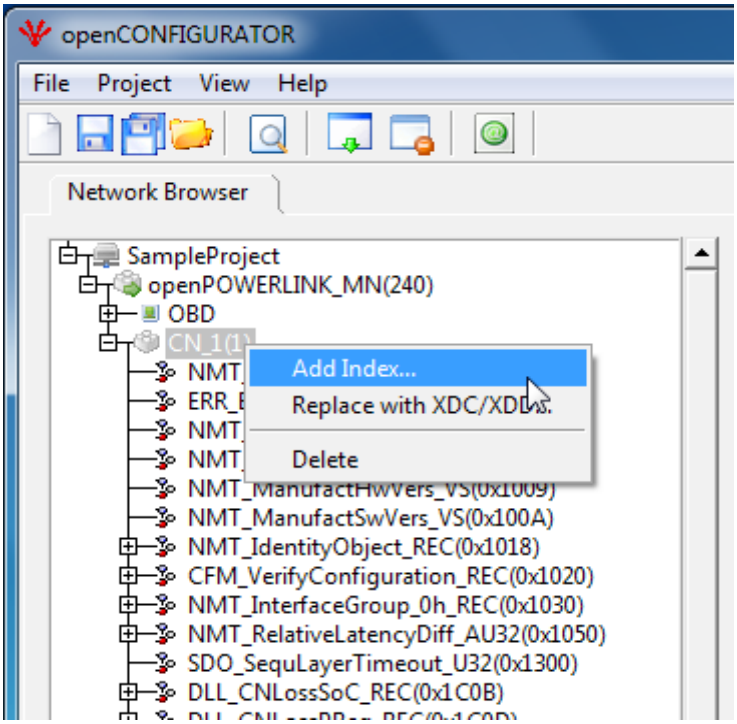


Illustration 16: Add Index Menu

- The 'Add Index' pop up window will appear asking for Index Id. Give the Index Id in hex (say 0x6000) and click 'Ok' to add the index

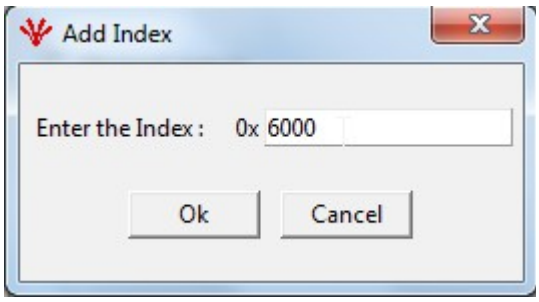


Illustration 17: Add Index Window

- The Index(0x6000) will be added to the Node(CN_1) as shown in the below illustration

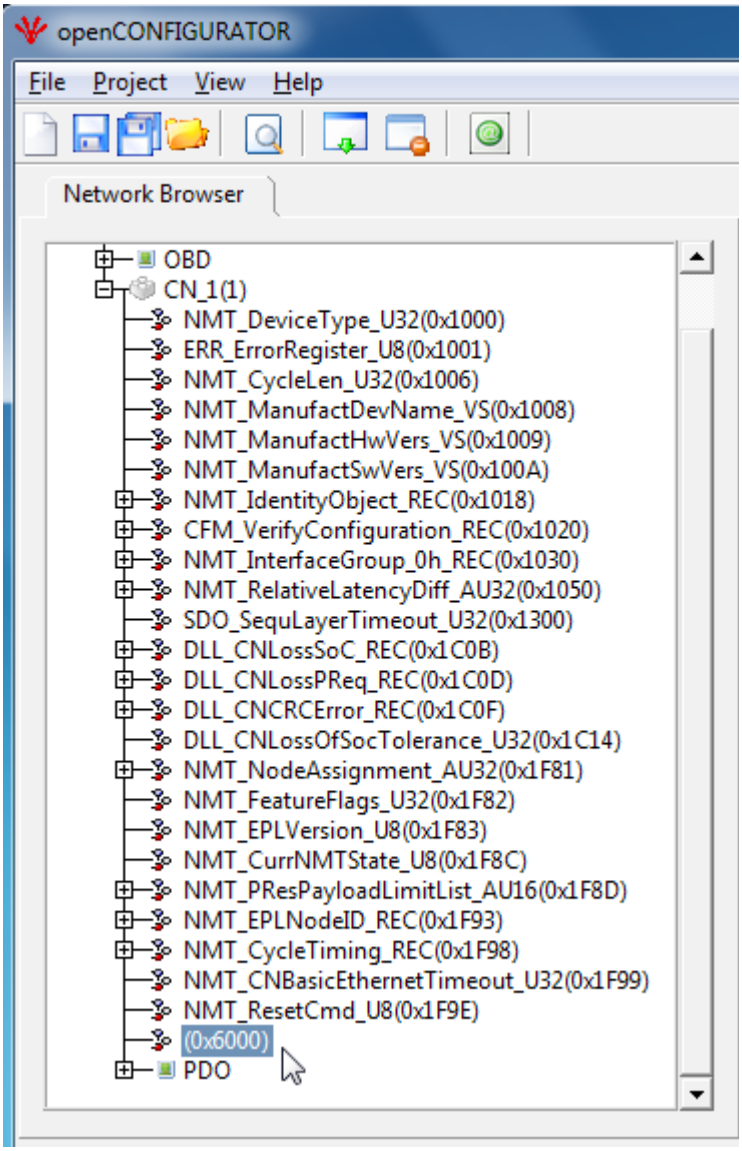


Illustration 18: Index Added - Tree

- Click on this newly added Index(0x6000) to display its 'Properties' on the right pane as shown below

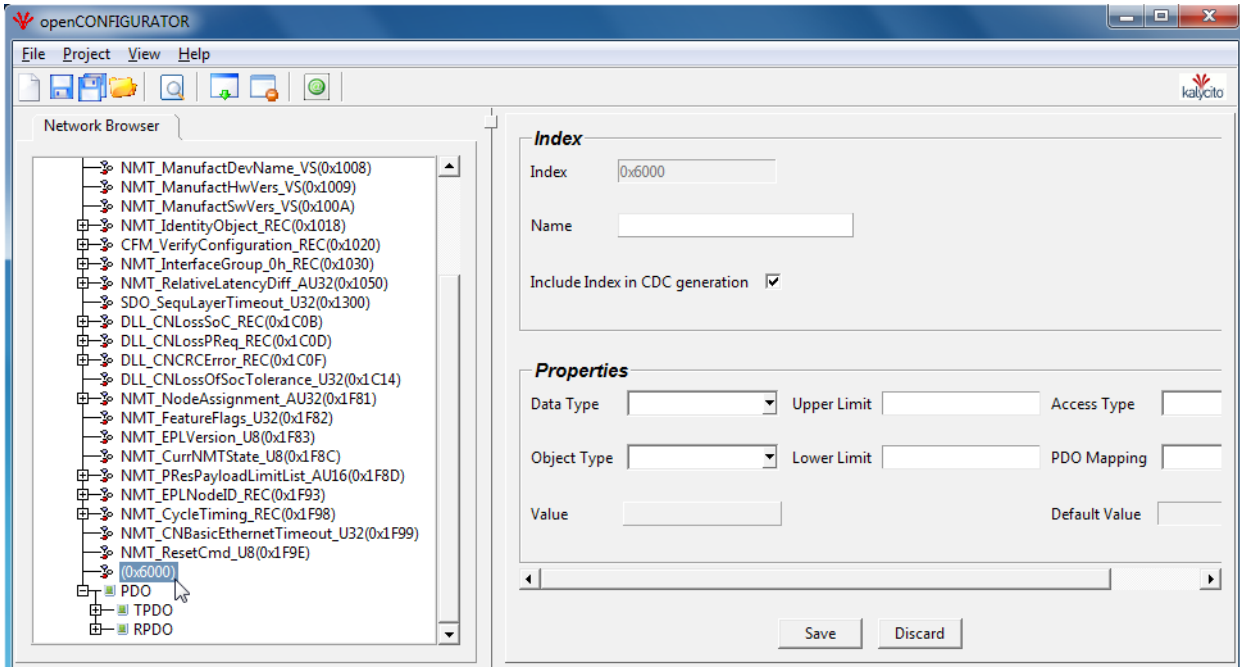


Illustration 19: Index Properties - Empty

Property	Description
Index*	Index Id
Name*	Name for the Index / SubIndex
Include In CDC generation	Determines the inclusion of the value of the Index in the CDC
DataType*	Data type of the object
ObjectType*	Object type of the object
AccessType*	Access rights for the particular object
PDO Mapping*	Determines the mapping type for the PDO object
Default Value	Default Value for the object
Value(Actual Value)	Actual Value for the object
Dec/Hex	Toggle between the decimal & hexadecimal view of the value
Upper Limit	Highest limit for the value of the object
Lower Limit	Lowest limit for the value of the object

* mandatory fields

- Now fill the values for the 'Properties' for the object/ subobject

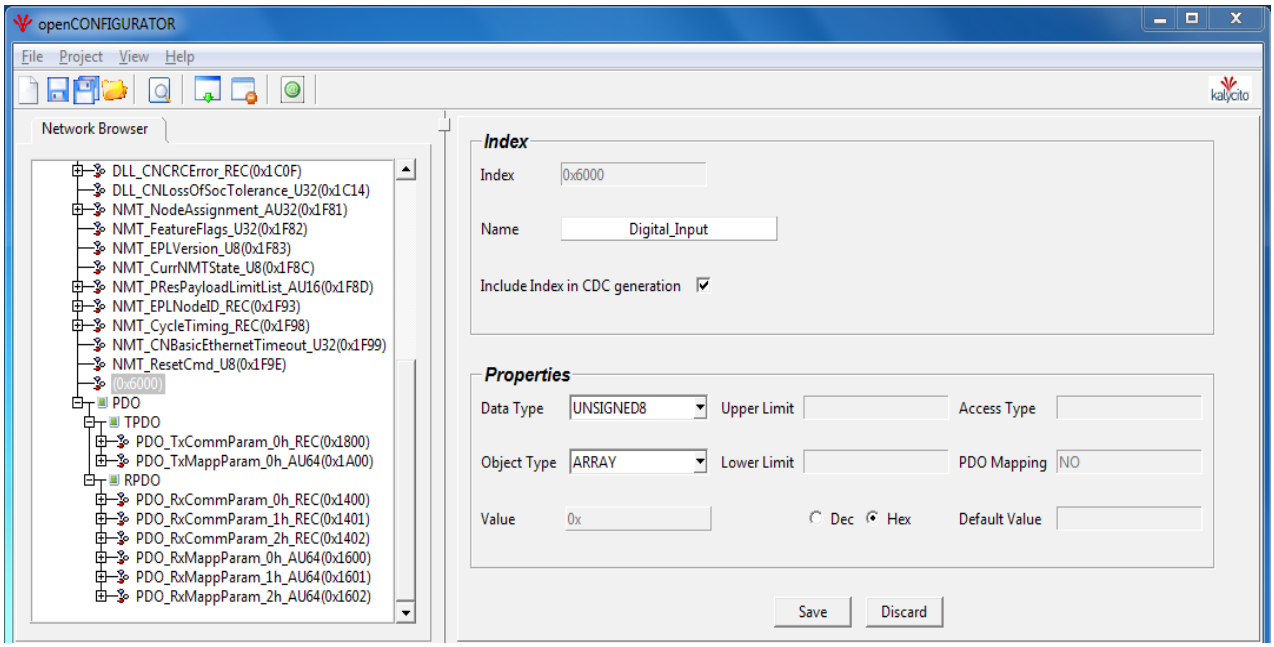


Illustration 20: Add Index Properties

- Choose the 'Object Type' as 'Array' and click 'Save'. The Index properties will be saved and the SubIndex with SubIndexId(0x00) will be created automatically as shown in the below illustration.

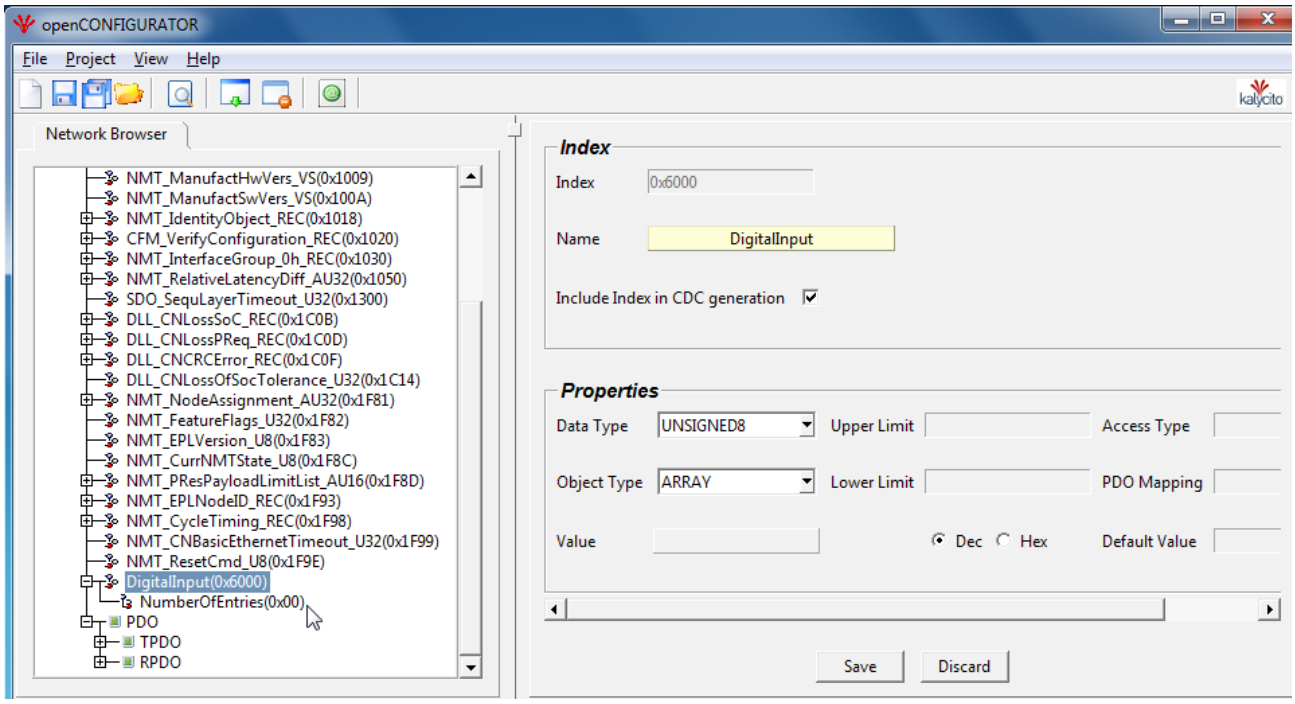


Illustration 21: Index Added

- To add a SubIndex for the output process variable, right click on the added process variable index (0x6000) and click "Add SubIndex" as shown in the below illustration

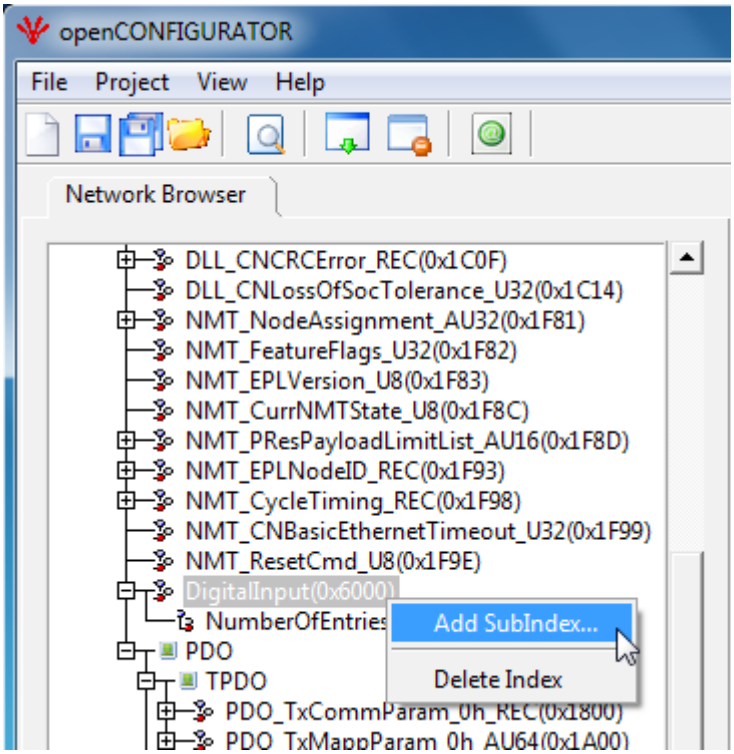


Illustration 22: Add SubIndex Menu

- A small pop up window opens asking for SubIndexId. Give the SubIndexId in hex(say 0x01) and press 'Ok' to add the SubIndex.

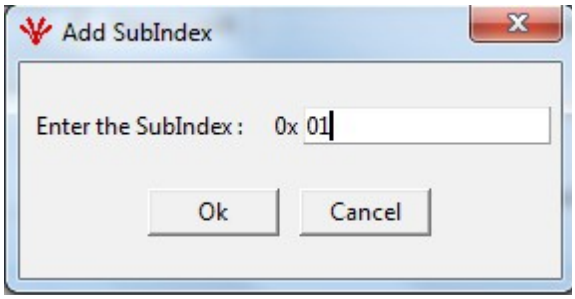


Illustration 23: Add SubIndex Window

- Then click on the newly added SubIndex. The 'Properties' will be empty as shown below

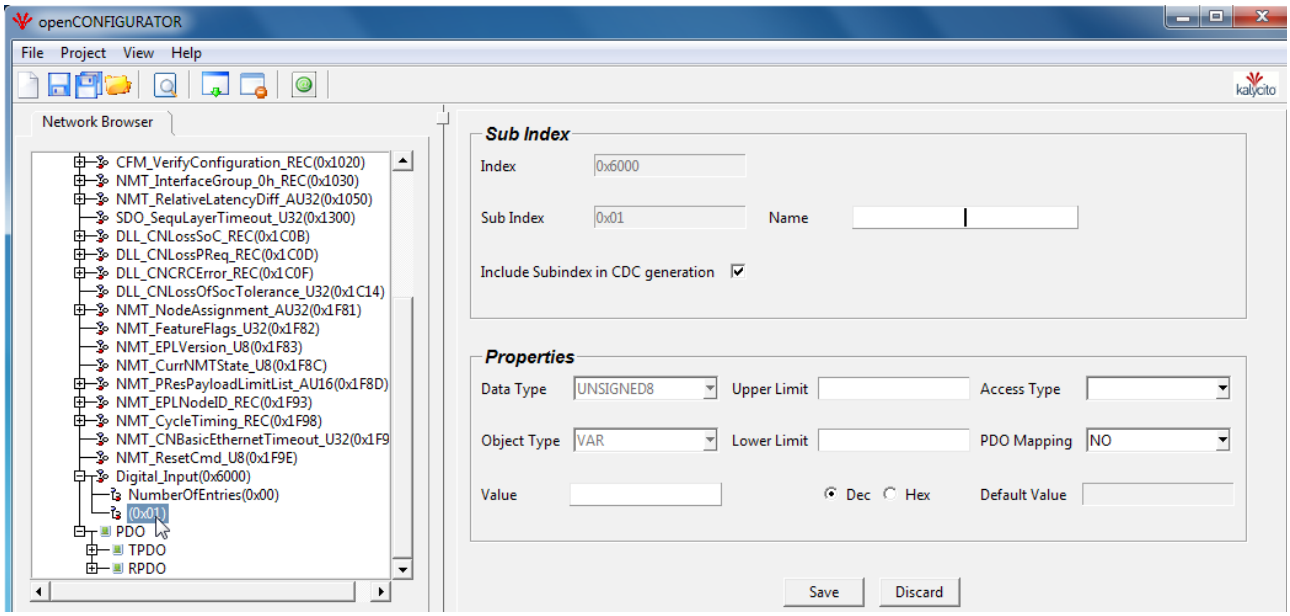


Illustration 24: Add SubIndex Properties- 1

- Then fill the 'Properties' on the right pane as shown in the below illustration and click 'Save'

Note:	Refer the property table in the section 3.3 for the property description

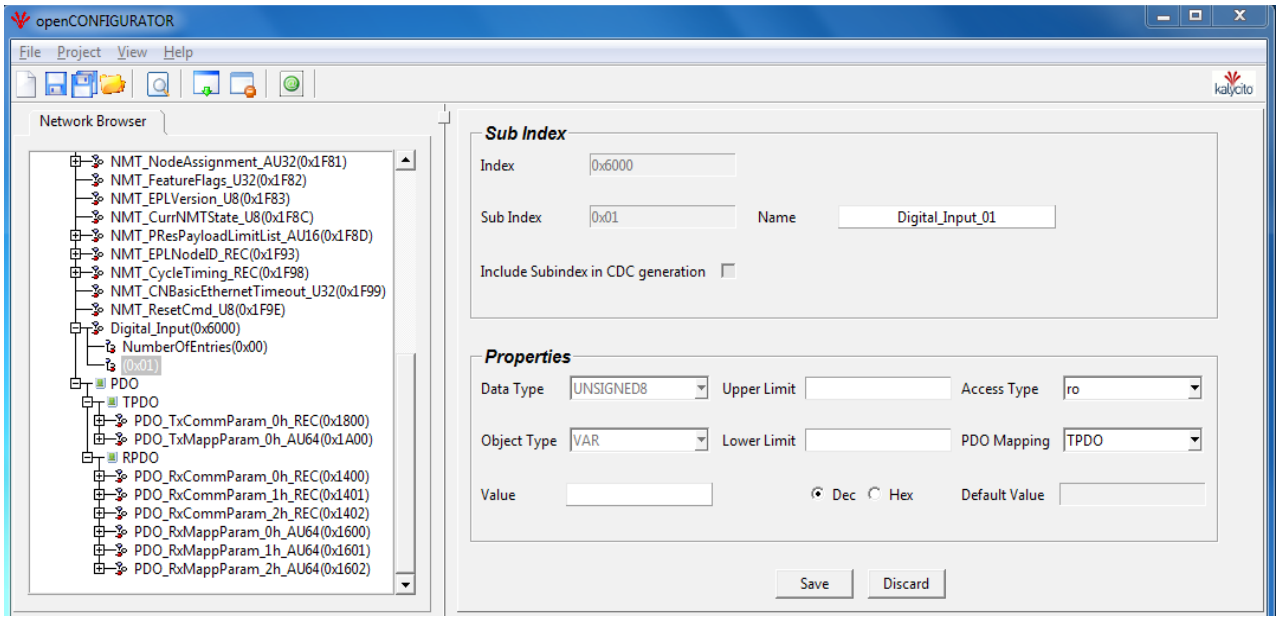


Illustration 25: Add SubIndex Properties- 2

- Similarly an input process variable can be added with IndexId(0x6200) and SubIndexId(0x01) as shown below

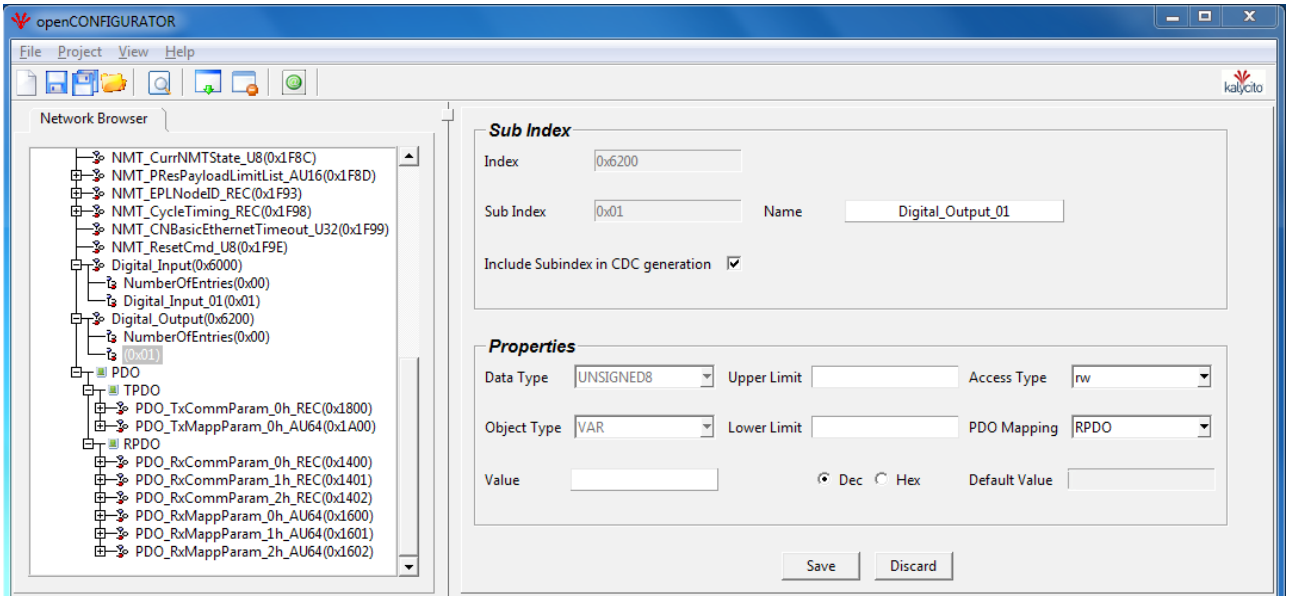


Illustration 26: Input PI variables

3.4 PDO Mapping for the CN process variables

A PDO(Process Data Object) is used to exchange process variables between the managing node(MN) and the controlled nodes(CN) of the Powerlink network. The two types of PDO are Receive PDO (RPDO) and Transmit PDO (TPDO).

The PDO table contains the Target Node Id, process variable Index, Sub Index, Length and Offset of the data in the payload. (For more information, refer section 6.4 of Ethernet POWERLINK Communication Profile Specification Version 1.1.0)

Mapping of PDO with the process variable on openPOWERLINK CN:
Click on TPDO from the "Tree Browser". A PDO mapping table will be loaded on the right pane as shown below

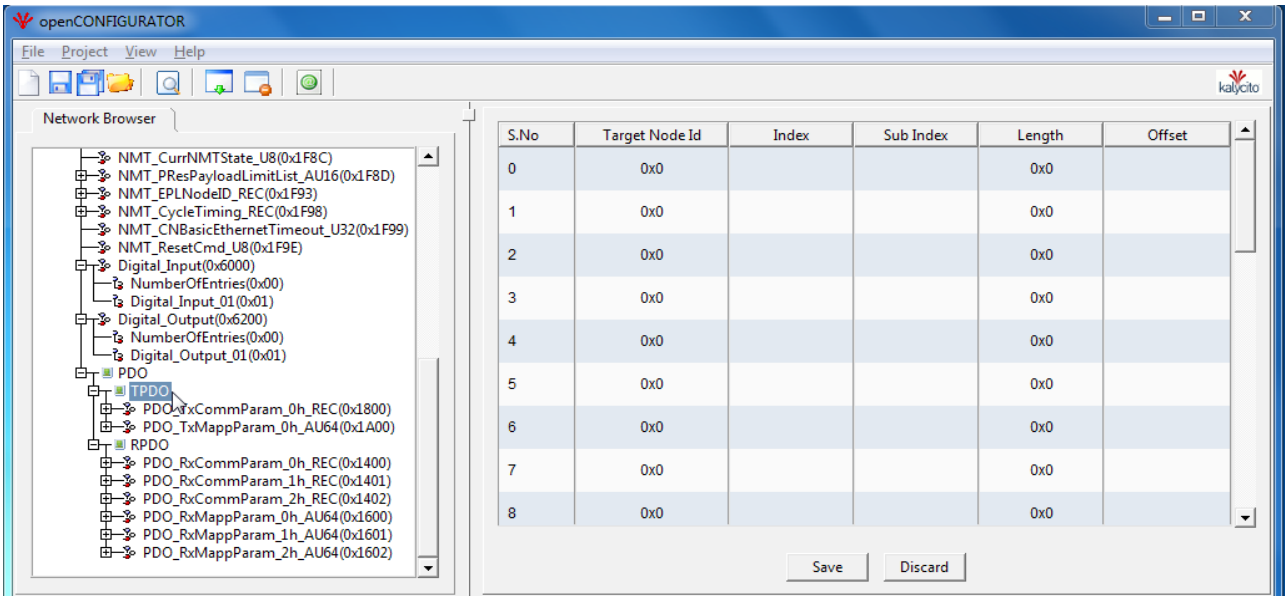


Illustration 27: PDO mapping table

Columns	Description	Allowed Range	
Target Node Id	Node Id of the PDO target	0x0	Broadcast Node Id
		0x1 – 0xEF	Available CN Node Id (used for cross traffic)
		0xF0	MN Node Id
Index	Index of the object to be mapped	0x1000 - 0x9FFF	Index/SubIndex which is allowed to be mapped as a PDO. Refer UserManual for more details
Sub Index	Sub-Index of the object to be mapped	0x00*, 0x01 – 0xFE	
Length	Length of the mapped object (Bit count)	Depends on the DataType property of the Index/SubIndex	
Offset	Offset related to the start of the PDO payload (Bit count)	Cumulative sum of the payload length	

- Select the appropriate 'Target Node Id' as shown in the below illustration.

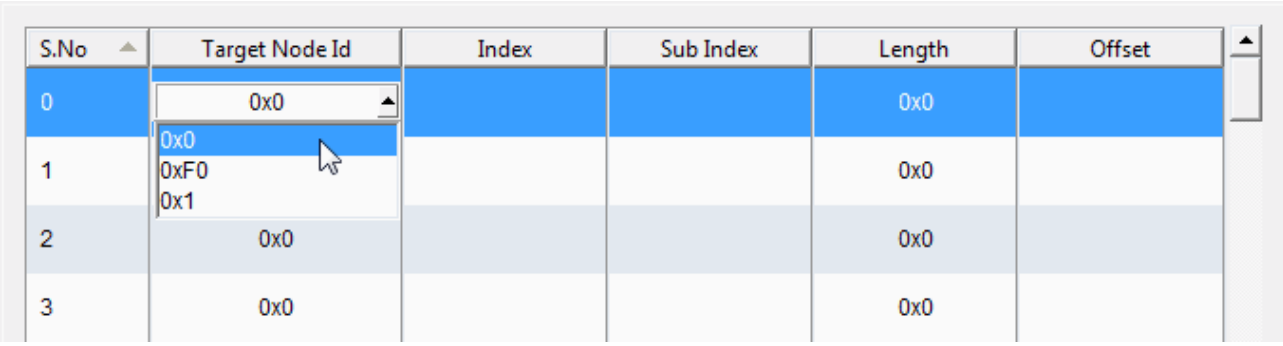


Illustration 28: PDO mapping table - Select Node id

- Select an 'Index' value available in that list

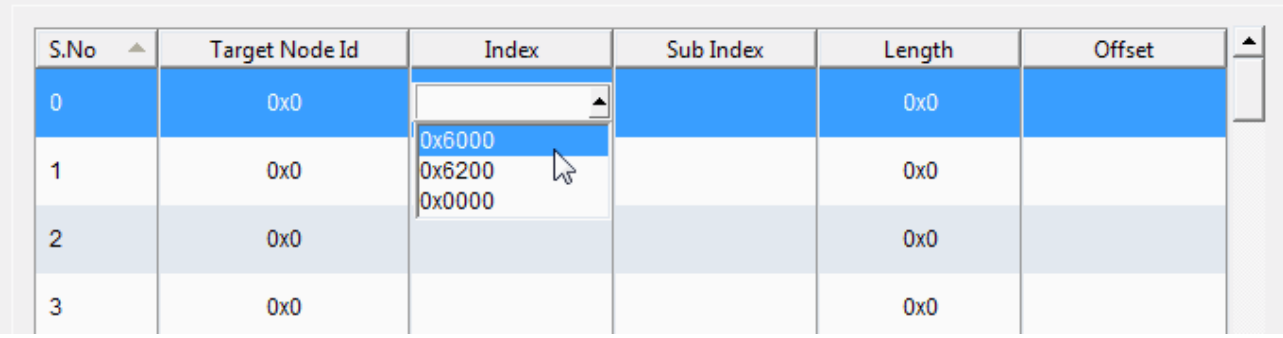


Illustration 29: PDO mapping table - Select Index Id

- Select a 'Sub-index' value within the range(0x00 – 0xFE)

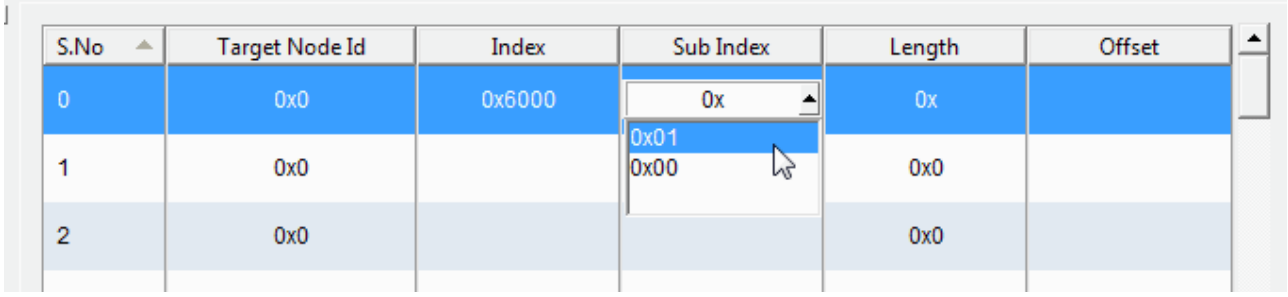


Illustration 30: PDO mapping table - Select SubIndex Id

- Select the 'Length' of the PDO object as shown below

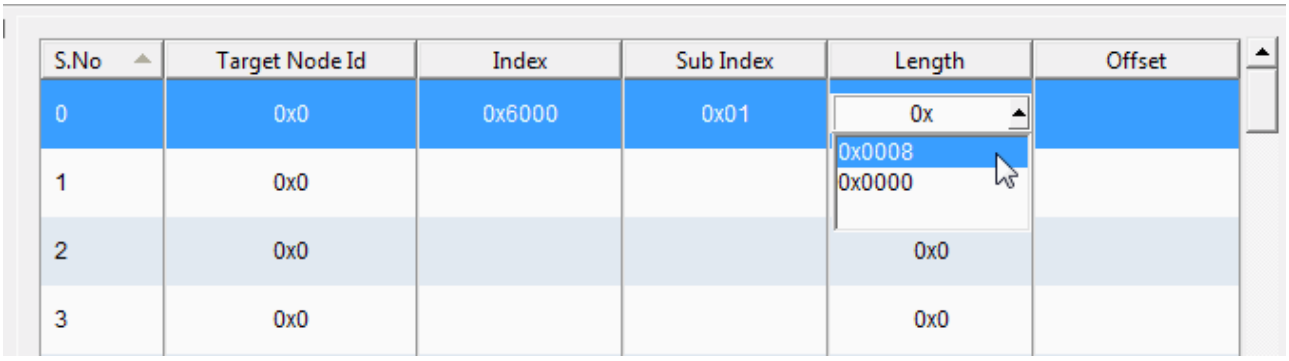


Illustration 31: PDO mapping table - Select Length

- The 'Offset' values will be updated automatically after the 'Length' is selected

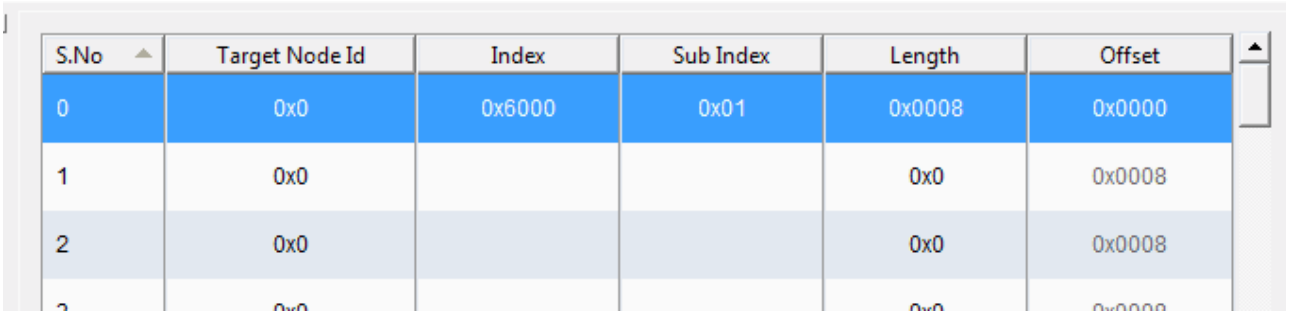


Illustration 32: PDO mapping table - Offset

- Save the changes by clicking on the 'Save' button or discard the changes by clicking the 'Discard' button

- Similarly, create the PDO mapping for the input process variable with the Index 0x6200(Digital_Output), SubIndex 0x01(Digital_Output_01) with 'Length' as 0x0008

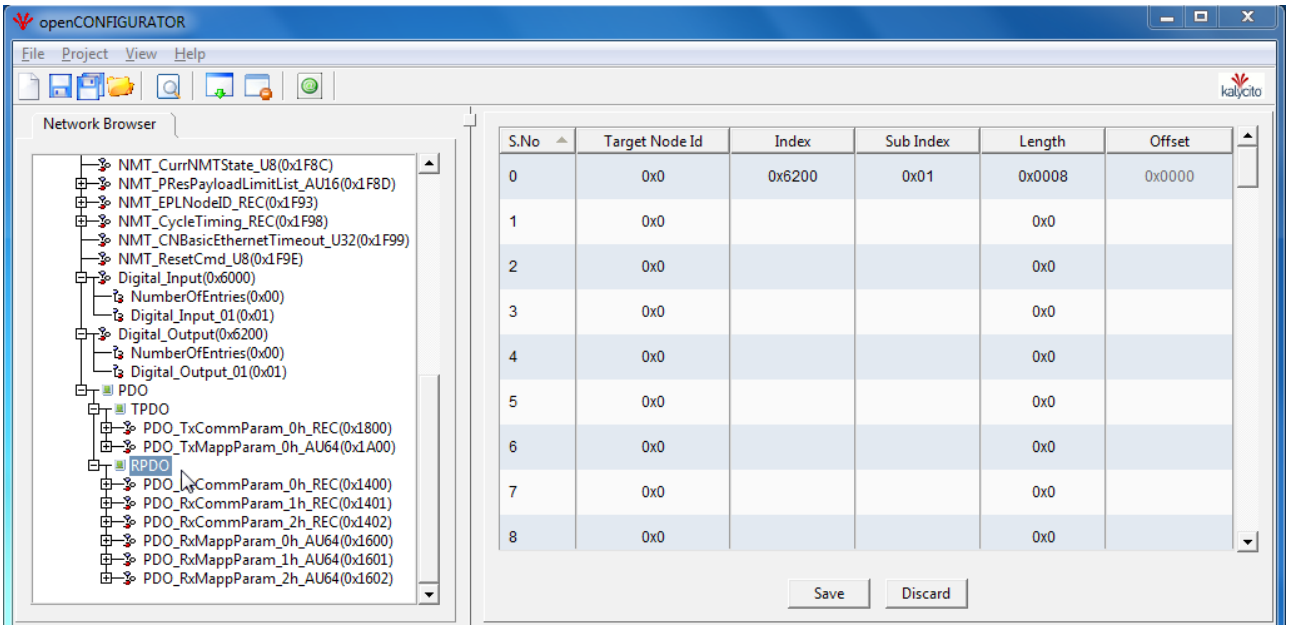


Illustration 33: PDO mapping table - RPDO mapping values

3.5 Build the sample project

- User can build the project by selecting Project > Build Project or by using the “F7” keyboard shortcut or by clicking on the build project icon

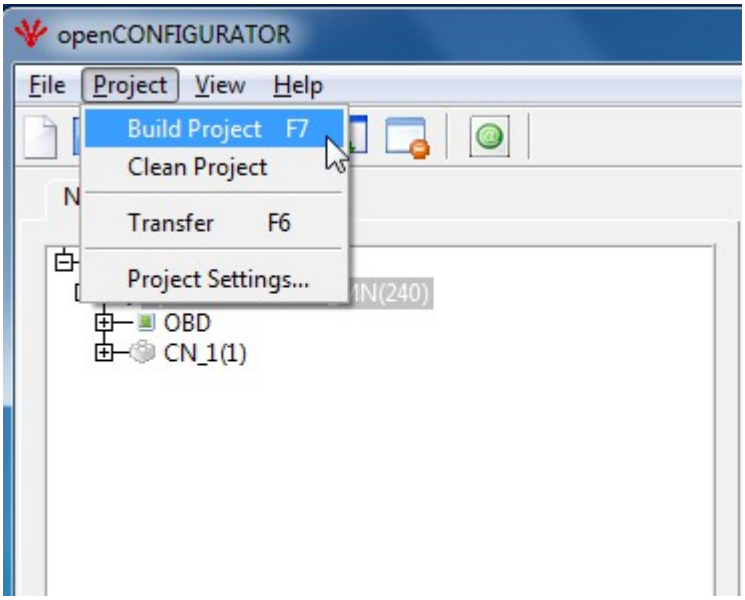


Illustration 34: Build Project Menu

- After selecting the 'Build Project' option, a message pop-up will indicate that user edited values for MN will be lost since PDO mapping will be calculated automatically.
- Click 'Yes' since Auto Generate MN OBD is enabled. This will automatically generate MN PDO mapping.

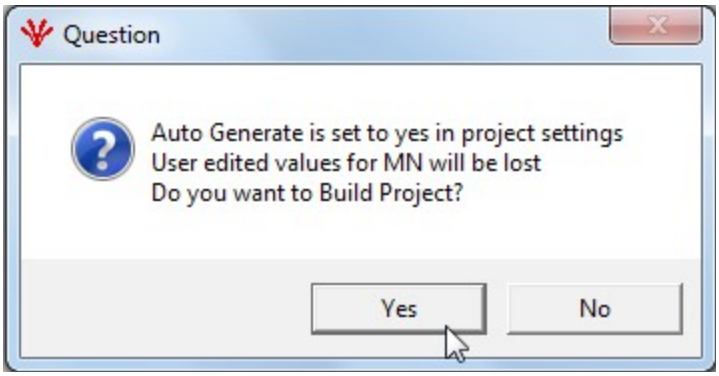
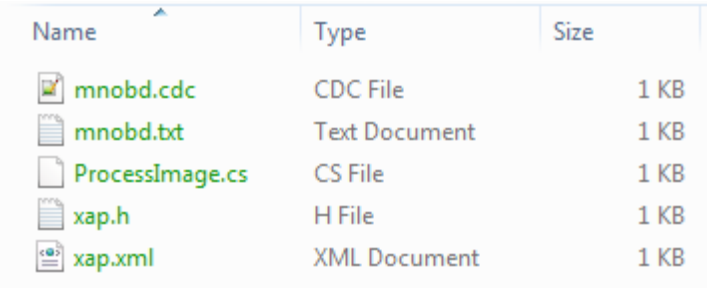


Illustration 35: Build Project - Auto Generate

Note:	Please refer the user manual for more details on Auto Generate 'No' option

- Following files will be created after the build of project. These files will be present in the absolute path <Project location >/<Project Name>/cdc_xap folder.



Name	Type	Size
mnobd.cdc	CDC File	1 KB
mnobd.txt	Text Document	1 KB
ProcessImage.cs	CS File	1 KB
xap.h	H File	1 KB
xap.xml	XML Document	1 KB

Illustration 36: cdc_xap Folder View

File name	Description
mnobd.cdc	CDC binary file used with the openPOWERLINK stack
mnobd.txt	Text version of the binary CDC file
XAP.h	Header file for the application
XAP.xml	XML file with the variables names, Datatype, Datasize, ByteOffsets, BitOffsets
ProcessImage.cs	A C# namespace with the application variables and the size of the data

mnobd.txt will be generated as below

```
00000012
//// NodeId Assignment
1F81    01      00000004        00000007

1600    00      00000001        00
1A00    00      00000001        00
1006    00      00000004        0000C350
1C02    01      00000004        00000028
1C02    03      00000004        00000028
1C09    01      00000004        00000028
1F26    01      00000004        00002A1E
1F27    01      00000004        04034C48
1F92    01      00000004        00006978
1400    01      00000001        01
1600    01      00000008        000800000001A4C0
1800    01      00000001        01
1A00    01      00000008        000800000001A040
1600    00      00000001        01
1A00    00      00000001        01
////Configuration Data for CN-1
1F22    01      00000079
0000000B
1600    00      00000001        00
1A00    00      00000001        00
1006    00      00000004        0000C350
1020    01      00000004        00002A1E
1020    02      00000004        04034C48
```

```
1C0B    03      00000004        00000050
1C0D    03      00000004        00000050
1600    01      00000008        0008000000016200
1A00    01      00000008        0008000000016000
1600    00      00000001        01
1A00    00      00000001        01
//// NodeId Reassignment
1F81    01      00000004        80000007
```

XAP.h will be generated as below

```
#ifndef XAP_h
#define XAP_h

# define COMPUTED_PI_OUT_SIZE 4
typedef struct
{
        unsigned CN1_M00_Digital_Input_Digital_Input_01:8;
        unsigned PADDING_VAR_1:24;
} PI_OUT;

# define COMPUTED_PI_IN_SIZE 4
typedef struct
{
        unsigned CN1_M00_Digital_Output_Digital_Output_01:8;
        unsigned PADDING_VAR_1:24;
} PI_IN;

#endif
```

XAP.xml will be generated as below

```
<?xml version="1.0" encoding="UTF-8"?>
<ApplicationProcess>
  <ProcessImage type="output" size="1">
    <Channel Name="CN1.Digital_Input.Digital_Input_01" dataType="Unsigned8" dataSize="8"
PIOffset="0x0000" BitOffset="0x00"/>
  </ProcessImage>
  <ProcessImage type="input" size="1">
    <Channel Name="CN1.Digital_Output.Digital_Output_01" dataType="Unsigned8" dataSize="8"
PIOffset="0x0000" BitOffset="0x00"/>
  </ProcessImage>
</ApplicationProcess>
```

ProcessImage.cs will be generated as below

```
using System;
using System.Runtime.InteropServices;

namespace openPOWERLINK
{

        /// <summary>
        /// Struct : ProcessImage Out
        /// </summary>
```

```
        [StructLayout(LayoutKind.Explicit, Pack = 1, Size = 4)]
        public struct AppProcessImageOut
        {
                [FieldOffset(0)]
                public byte CN1_M00_Digital_Input_Digital_Input_01;
                [FieldOffset(1)]
                public byte PADDING_VAR_1;
                [FieldOffset(2)]
                public byte PADDING_VAR_2;
                [FieldOffset(3)]
                public byte PADDING_VAR_3;
        }

        /// <summary>
        /// Struct : ProcessImage In
        /// </summary>
        [StructLayout(LayoutKind.Explicit, Pack = 1, Size = 4)]
        public struct AppProcessImageIn
        {
                [FieldOffset(0)]
                public byte CN1_M00_Digital_Output_Digital_Output_01;
                [FieldOffset(1)]
                public byte PADDING_VAR_1;
                [FieldOffset(2)]
                public byte PADDING_VAR_2;
                [FieldOffset(3)]
                public byte PADDING_VAR_3;
        }
}
```

4 Conclusion

The CDC, XAP & C# namespace generated after build process can be used in the openPOWERLINK project for running the application.

5 Appendix – Abbreviations

MN	Managing Node
CN	Controlling Node
CFM	Configuration Manager
PI	Process Image
OBD	Object Dictionary
PDO	Process Data Object
CDC	Concise Data Configuration
XDD	XML Device Description
XDC	XML Device Configuration
XAP	XML Application

6 Support

Please post your queries and suggestions on the appropriate topic in the openCONFIGURATOR discussion forum at Sourceforge.